ESTRATEGIA DE SELECCIÓN DE ENTRADAS Y PARÁMETROS ÓPTIMOS PARA MÁQUINAS DE SOPORTE VECTORIAL

Strategy of selecting optimal inputs and parameters for support vector machines

RESUMEN
Los métodos de clasificación basados en el conocimiento (MCBC), en especial las máquinas de soporte vectorial son una metodología que en la práctica presentan excelentes resultados. Estos métodos de clasificación requieren para su buen funcionamiento una selección rigurosa de entradas (descriptores) y una adecuada calibración de parámetros. El espacio de búsqueda para la selección de entradas y el ajuste parámetros es muy grande por lo que en este artículo se presenta un algoritmo de optimización para la selección de entradas y ajuste de parámetros óptimo para las máquinas de soporte vectorial. Los resultados obtenidos en casos de prueba de la literatura especializada son satisfactorios para el algoritmo de optimización presentado.

PALABRAS CLAVES: búsqueda en vecindario variable, máquinas de soporte vectorial, redes de Kohonen.

DAVID ÁLVAREZ MARTÍNEZ
Ingeniero de Sistemas y Computación,
Joven Investigador
COLCIENCIAS-Universidad Tecnológica de Pereira
davidalv@utp.edu.co

GOBER RIVERA MONROY
Ingeniero de Sistemas y Computación,
Universidad Tecnológica de Pereira
gober@utp.edu.co

JUAN JOSE MORA
Ingeniero Electricista, Ph.D.
Docente
Programa Ingeniería Eléctrica
Universidad Tecnológica de Pereira
jjmora@utp.edu.co

ABSTRACT
The classification methods based on knowledge (MCBC) in particular support vector machines are a methodology that, in practice, have excellent results. These machines require of a rigorous selection of the inputs (descriptors) and the calibration parameters for the proper functioning. The search space for the selection of input and calibration parameters is very large and that is the reason of why in this paper presents an optimization algorithm for the selection of inputs and parameters. The results obtained in test cases of the literature specialized are satisfactory for the optimization algorithm presented.

KEYWORDS: *Kohonen, support vector machine, variable neighborhood search.*

1. INTRODUCCIÓN

En general, los conocimientos de los que se dispone para abordar un problema particular son de tipo empírico y teórico y forman conjuntos que se interceptan o se complementan. En la mayoría de casos, los conocimientos disponibles sobre un problema en particular, no son totalmente correctos ni completos; por esta razón la información disponible debe explotarse de la mejor forma posible para entender mejor los fenómenos que representan.

Los métodos de clasificación basados en conocimiento (MCBC), utilizan información empírica que esté declarada o explicita. Uno de los métodos más usados son las máquinas de soporte de vectorial (MSV), que son una metodología que en problemas de clasificación presentan resultados muy satisfactorios, pero que como muchos los métodos requieren de una adecuada parametrización, por lo cual se deben seleccionar las entradas y ajustar los parámetros, para así alcanzar resultados óptimos con la metodología.

En la literatura especializada no se encuentra una técnica eficiente para dicho problema, por lo cual como aporte a la solución de este problema, en este articulo se presenta una metodología que hace una propuesta a la parametrización que tiene la MSV, permitiendo seleccionar las entradas y ajustar los parámetros de penalización (C) y kernel Gaussiano (σ) para obtener mejores resultados en la clasificación.

Como contenido del artículo, en la sección dos se presenta la definición del problema de selección de entradas y ajuste de parámetros para una MSV, en la sección tres se plantea el algoritmo de optimización propuesto, mientras que en la sección cuatro se muestran los casos de prueba. A continuación, en la sección cinco se presentan los resultados comparando las respuestas con un modelo de calibración de parámetros y selección de entradas basado en permutaciones de entradas y búsqueda exhaustiva de parámetros en malla contra el modelo de optimización propuesto y finalmente en la sección seis las conclusiones y perspectivas de trabajos futuros.

Fecha de Recepción: 26 de enero de 2009
Fecha de Aceptación: 12 de mayo de 2009

2. SELECCIÓN DE ENTRADAS Y AJUSTE ÓPTIMO DE PARAMETROS DE UNA MSV

Los métodos de clasificación basados en el conocimiento (MCBC), utilizan información declarada o explicita en bases de datos históricas con características significativas o patrones para determinar la solución del problema. Las MSV son una metodología que cumplen con las propiedades mencionadas anteriormente. Éstas fueron desarrolladas por Vladimir Vapnik usando herramientas estadísticas, de optimización y de las máquinas de aprendizaje [2]. Antes de que Vladimir Vapnik desarrollara el modelo de programación no lineal, Olvi L. Mangasarian había desarrollado un modelo de programación lineal para la separación de patrones lineales y no lineales [3], este trabajo es reconocido en el campo de MCBC. La arquitectura de las MSV sólo depende de un parámetro de penalización denotado como C que puede ser visto como un parámetro para controlar el sobre-entrenamiento y la función *kernel* (incluyendo sus parámetros). En el caso de la función de base radial (RBF), existe sólo un parámetro a configurar denotado como σ [4]. Una metodología interesante para la parametrización de las MSV es presentada en [5], en este el parámetro C es cambiado por nuevo parámetro v y una variable ρ para ser optimizada, aunque v es un parámetro definido en un intervalo pequeño, igual se debe realizar un paso de optimización para la variable ρ.

En [1] las MSV demuestran su fortaleza como MCBC, creando un importante campo de estudio a la parametrización y selección de entrada óptima de las MSV.

La selección óptima de entradas de una MSV tiene un espacio de búsqueda igual a 2^n, donde n es el número de entradas o descriptores preseleccionados. El ajuste de parámetros de una MSV tiene un espacio de búsqueda que en [1] se determinó empíricamente como $Cmin<C<Cmax$ y $\sigma min<\sigma<\sigma max$ los valores de $Cmin$ y $Cmax$ son 2^4 y 2^{32} respectivamente, de otra parte, los valores de σmin y σmax son 2^{-6} y 2^6.

Como la selección de entradas guarda relación con los parámetros seleccionados, ajuste parámetros y selección de entradas óptimas comparten el mismo espacio de búsqueda, el cual sería para el caso de este artículo 2^n por un número real entre 2^4 y 2^{32} por un número real entre 2^{-6} y 2^6. Obteniendo un espacio de soluciones infinito o si los valores de C y σ, se restringe para tomar valores enteros se tiene un espacio de soluciones muy grande.

Analizando el espacio de soluciones del problema propuesto en este trabajo y teniendo como función objetivo maximizar la calidad de clasificación de la MSV, se observa claramente que el uso de una técnica de optimización es necesario para su solución.

La figura 1 representa la arquitectura de componentes que actúan sobre el proceso de optimización de la máquina de soporte vectorial.

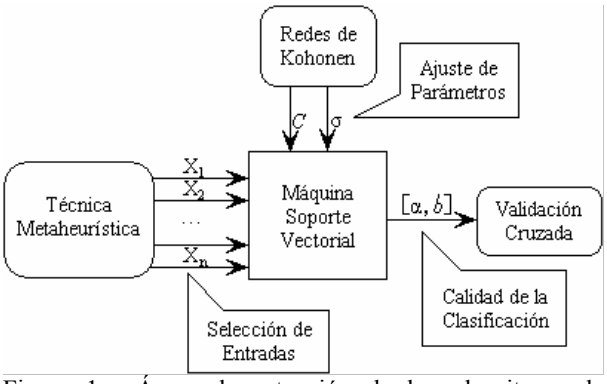

Figura 1. Áreas de actuación de los algoritmos de optimización.

3. ALGORITMO DE OPTIMIZACIÓN PROPUESTO

3.1 **Codificación**

Sea X una n-tupla de la siguiente forma $X = (x_1, x_2, ..., x_i, ..., x_n)$ y x_i una variable binaria que toma el valor de 1 si la entrada i esta activa en la MSV o 0 si no lo está, sea C, una variable real en un intervalo de solución de $[2^4, 2^{32}]$ y sea σ, una variable real en un intervalo de solución igual a $[2^{-6}, 2^6]$. Con esto tenemos nuestras variables del problema codificadas y restringidas en un intervalo.
La codificación de las variables propuesta anteriormente permite que cualquier técnica de optimización pueda ser usada, como las variables X, C y σ son independientes entre sí, para cada conjunto de valores de X existe una valuación C y σ que representa la mejor calidad de clasificación con el vector X como entrada a la MSV. De esta forma se puede utilizar cualquier algoritmo combinatorial para encontrar valores óptimos de X siempre y cuando se utilice un segundo algoritmo que permita encontrar los valores óptimos partir de los valores de C y σ.

3.2 **Algoritmo híbrido de búsqueda en vecindad variable y Kohonen**

La figura 2 esquematiza el proceso de optimización a realizarse.

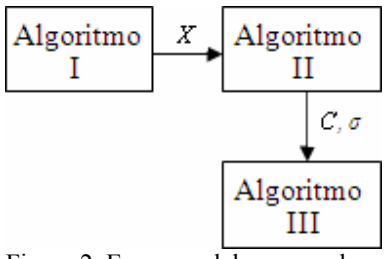

Figura 2. Esquema del proceso de optimización

3.2.1 Algoritmo de búsqueda aleatoria binaria

El algoritmo I determina el valor de la variable binaria X_j la cual representa si el descriptor es tomado en cuenta o no, para este algoritmo se utilizó un esquema de vecindario variable [6, 7].

Algoritmo I: Búsqueda aleatoria binaria

```
Datos de entrada ← IteracionesGlobales
Xinc ←Aleatorio Binario
Incumbente ← 0
Vecindario ←1
For k = 1 To IteracionesGlobales
    X←Xinc
    For i = 1 To Vecindario
        j ←Aleatorio Entero
        X[j] ←Not(X[j])
    EndFor
        Vecindario ←Vecindario + 1
        F(c, σ) ←AlgoritmoII(X)
        If F(c, σ) > Incumbente
                Incumbente ← F (c, σ)
                Vecindario ← 1
        EndIf
    EndFor
EndFor
Resultado ←Xinc, c, σ
```

Inicialmente el vecindario de búsqueda consiste en modificar de forma aleatoria una posición en el vector *X*. Posteriormente, si esta modificación no genera una mejora en la función objetivo se procede a modificar dos posiciones y así sucesivamente, en el momento en que la incúmbete sea actualizada se regresa nuevamente al primer vecindario. Esta metodología de búsqueda local permite diversificar cuando la función objetivo no es mejorada y hacer una búsqueda local más detallada en el momento de actualizarse la incumbente; de esta forma se logra un algoritmo computacionalmente eficiente.

3.2.2 Algoritmo de Kohonen.

El segundo algoritmo determina las variables *C y σ* a partir un vector *X*, para ello utiliza los mapas auto-organizados de Kohonen [8, 9, 10]. Que modifica iterativamente el valor la variable *C y σ*. Se requiere definir una tasa de aprendizaje, un numero de iteraciones y valores iníciales para la función de vecindad (por general Gaussiana), la técnica construye una malla sobre los datos haciendo que cada uno de ellos apunte a los nodos de dicha malla, después se ingresa un vector inicial P_j y se calcula la mínima d_i distancia (distancia euclidiana) entre el vector y cada uno de los nodos optimizando también la función objetivo (validación cruzada). Como se puede observar los nodos o neuronas de la capa de salida "compiten" por activarse, el algoritmo entonces retorna las coordenadas sobre el espacio de la neurona ganadora, además de esto se actualizan los pesos de las neuronas vecinas perdedoras de acuerdo a los esquemas de vecindad propuestos (función de vecindad).

Las entradas se normalizan y se procesan una por una, es decir se ubican dentro de una nueve de datos y se asignan a la malla inicial. La aplicación a problemas de optimización como este se hacen normalmente escogiendo una topología en los nodos de salida que refleje la posible solución del problema a resolver. En este caso hay trabajar sobre una nube de datos lo que tendremos al final es una cantidad de mínimos locales como nodos en la malla existan.

Cabe anotar que para cada iteración la configuración de la malla es distinta y cuando se una nueva iteración del problema la configuración de la malla final no va hacer igual a la de iteraciones pasadas, ni van a tender a una configuración de malla final.

Algoritmo II: Mapas Auto-Organizados de Kohonen

Inicializar aleatoriamente los pesos con valores bajos de las *N* entradas a la *M* salidas
Dada una nueva entrada:

1) Calcula la distancia entre la entrada y casa nodo de salida j (1)

$$d_j = \sum_{i=0}^{n-1} (X_i(t) - W_{ij}(t))^2 \qquad (1)$$

Donde $X_i(t)$ es la *i*-ésima entrada al tiempo *t* y $W_{ij}(t)$ es el peso del nodo ***i*** al nodo de salida ***j.***

2) Selecciona el nodo con la distancia menor (***j****) y que además maximice la validación cruzada.

3) Actualiza los pesos del nodo j* y los de sus vecinos de acuerdo al esquema de vecindad (2).

$$W_{ij}(t+1) = W_{ij}(t) + \propto(t)\left(X_i(t) - W_{ij}(t)\right) \qquad (2)$$

Donde $\propto(t) (0 < \propto(t) < 1)$ decrece con el tiempo.

El algoritmo toma como valor de cada nodo el valor que optimiza la validación cruzada, para lo cual se mueve alrededor de la vecindad buscando el que mejor resultados obtenga de dicha validación.

3.2.3 Algoritmo de validación cruzada para la ponderación del grado de exactitud de la detección de faltas

El tercer algoritmo determina el porcentaje ponderado de exactitud de clasificación de la MSV con las entradas ***X*** y los parámetros *C y σ* mediante la técnica de validación cruzada.

Algoritmo III: Porcentaje de Exactitud Ponderada de Validación Cruzada de la Maquina de Soporte Vectorial.

Datos de entrada ← *X*, σ, *c*, *DatosEntrenamiento*
For i=1 **to** NúmeroSubconjuntosDE
PorcentajeExactitud(i)←CalcularPEMSV
EndFor

$$\text{Resultado} \leftarrow \frac{\sum_{i=1}^{NSDE} Porcentaje \quad Exactitud \quad (i)}{Numero \quad Subconjunt \quad os \quad DE}$$

La validación cruzada pretende maximizar el porcentaje de aciertos que tenga los casos de prueba y validación que hace la MSV. Realiza una ponderación de dichos casos satisfactorios durante las pruebas y retorna el valor resultan de de la configuración de la MSV según la selección de entradas y el ajuste de parámetros resuelto con anterioridad.

Una representación de dicha validación se hace a continuación (ver figura 3):

V = Datos Validación
P = Datos Prueba
E = Exactitud o Calidad de la Clasificación Promedio
E'_i= Exactitud de la validación *i*

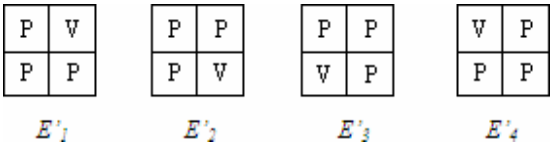

Figura 3. Esquema de validación para 4 subconjuntos.

Donde *E* es la suma de los E'_i encontrados sobre *N*, donde *N* es el número en el cual se dividieron los datos de prueba y de entrenamiento (en nuestro caso 4).

4. IMPLEMENTACIÓN

Todos los algoritmos descritos anteriormente fueron implementados en MATLAB®, usando el optimizador *QP* desarrollado por [11], la ToolBox de redes neuronales de MATLAB® para el algoritmo de Kohonen y el algoritmo de vecindad variable fue implementado como aporte al artículo.

Los casos de estudio son datos prácticos (*datasets)* tomados de [12], estos presentan diferentes problemas de clasificación con diversos números de descriptores y gran cantidad de datos, esto nos brinda que sean un buen conjunto de problemas de prueba con un nivel de complejidad aceptable para trabajo que se desarrolla en este artículo. Estos *datasets* son usados por la literatura especializada para probar y comparar metodologías de clasificación.

Las pruebas se realizaron en una máquina con un procesador de 2.2 GHz y una memoria RAM de 1Gb. Durante las pruebas no fue tomado en cuenta el factor tiempo, debido a que el tiempo invertido en el proceso de optimización depende realmente del tamaño de los datos de entrenamiento.

5. ANALISIS DE RESULTADOS

La metodología utilizada se comparo contra una selección de entradas por permutaciones y un ajuste de parámetros por búsqueda exhaustiva en malla. Los 7 casos de prueba fueron tomados de [12] y la identificación de los descriptores no se conoce, por lo cual se toman como un número. Los resultados se presentan en la tabla 1.

A partir de los resultados, no se puede asegurar que el mejor resultado de la red sea el óptimo global del ajuste de parámetros. Las redes neuronales de Kohonen fueron implementas como optimizadores.

Se observa en la tabla 1 que los porcentajes de exactitud o calidad de clasificación, dado por la selección de entradas y la parametrización de la metodología propuesta, en todos los casos de pruebas iguala en calidad de respuesta y en el caso de las instancias Glass y Pima se superan estos valores.

	Permutaciones y Búsqueda en Malla				**Metodología Hibrida Propuesta**			
Caso de Prueba	**Descriptores**	**Exactitud**	**C**	σ	**Descriptores**	**Exactitud**	**C**	σ
British	1,2,4	0,861818	256	16	1,2,4	0,861	16777216	0,03125
Fosil	2,4,6	0,7426	4092	0,0625	1,2,3,4,6	0,742	8388608	0,125
Gauss2d	1,2,	0,6412	1048576	4	1,2,	0,612	99454847	0,04878
Glass	1,2,	0,5643	268435456	0,5	1,2,	0,653	10054857	3,6564
Iris	1,2,	0,6145	32768	32	1,2,	0,614	784445	6,4848
Iris1	4,5,9	0,7952	2147483648	32	3,5,6,7,8	0,795	87854121	58,15451
Pima	4,5,7	0,615	16	0,015625	2,3,6,7,8,	0,618	96454163	22,456451

Tabla 1. Resultados Obtenidos Permutaciones y Búsqueda en Malla y la Metodología Hibrida Propuesta.

6. CONCLUSIONES

En esta investigación se realizó una selección de entradas y un ajuste de parámetros óptimos mediante técnicas que carecen de parametrización, obteniendo unos resultados iníciales para el estudio de este problema. Adicionalmente se implementó el algoritmo de vecindario variable para el problema de selección de entradas y mapas auto-organizados de Kohonen para el ajuste de parámetros, de una MSV. A partir de esta implementación se presentan muy buenos resultados, a partir de la interacción entre esas técnicas.

Adicionalmente, el uso de una técnica de *Soft Computing* como optimizador puede ser el mayor aporte de este trabajo. Se aprecia que los mapas Auto-Organizados de Kohonen para el problema de ajuste parámetros de una MSV, se comporta satisfactoriamente, no siendo la mejor elección ya que en general las redes neuronales no garantizan encontrar una solución optima.

Finalmente, los autores recomiendan atacar el problema de selección de entradas y ajuste de parámetros óptimos mediante técnicas puras de optimización, como propuesta se presenta trabajar nuevamente la selección de entradas mediante el algoritmo de vecindario variable y el ajuste de parámetros a través de búsqueda aleatoria real basada en recocido simulado. Estas dos se proponen porque el fin del problema es parametrizar y no se deben utilizar metodología que deban ser parametrizadas ya que así se traslada el mismo problema a la técnica usada para parametrizar.

7. BIBLIOGRAFÍA

[1] J. Mora, *Localización de Faltas en Sistemas de Distribución de Energía Eléctrica usando Métodos Basados en el Modelo y Métodos Basados en el Conocimiento*, Tesis doctoral. Universidad de Girona, España, 2006.

[2] V. Vapnik, *The nature of Statistical Learning Theory*, Second Edition, Springer Verlag, 2000.

[3] O. L. Mangasarian. "Linear and nonlinear separation of patterns by linear programming". *Operations Research*, vol. 13, 1965, pp.444–452.

[4] C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery, 1998, pp. 121-167.

[5] A. Smola, B. Schölkopf, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[6] Hansen P. Nenad M. Moreno J. "Búsqueda de entorno variable". *Revista Iberoamericana de Inteligencia Artificial*. vol.19, pp. 77-92, 2003.

[7] R. Gallego, A. Escobar, R. Romero, *Técnicas de optimización combinatorial*, Universidad Tecnológica de Pereira. 2006.

[8] Angel Kuri, *Redes de Kohonen y la Determinación Genética de las Clases*, Instituto Tecnológico Autónomo de México, Octubre de 2001.

[9] A. Sánchez, J.C. Moctezuma, S. Sánchez, *Redes Neuronales de Hopfield y Kohonen para la solución del problema del agente viajero*", Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación. 2005.

[10] F. J. Ribadas, (2005, Dec.). *Mapas auto-organizativos*, [Online]. Available: http://ccia.ei.uvigo.es/docencia/IA/Tema5_parte2.pdf

[11] Steve Gunn, srg@ecs.soton.ac.uk, (2008, Dec.). University of Southampton, [Online]. Available: http://users.ecs.soton.ac.uk/srg/softwaretools/

[12] Juan Méndez. jmendez@dis.ulpgc.es, (2008, Dec.). Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería, [Online]. Available: http://serdis.dis.ulpgc.es/~ii-rf/DataSets/